# **赫夫曼编码和解码**

# 目的

1. 掌握二叉树的顺序存储、遍历等基本知识。

2. 了解并掌握二叉树的非递归遍历方法。

3. 用最优二叉树解决具体应用问题。

# 问题描述

实现教材上赫夫曼编码的编码和译码方法。

# 步骤

1. 设计赫夫曼编码与译码算法。

2. 使用课件或教材中的例子进行测试和验证。

3. 对一段文字进行编码和译码处理，分别输出编码结果和译码结果。

# 设备和环境

PC 计算机、Windows 操作系统、C/C++开发环境

# 结论

能够理解和掌握二叉树的加权路径长度并用来进行字符的前缀编码。

思考

如何处理码表之外的字符？

实现要求

1. 命令行中运行编解码程序 huffman.exe 或者 ./huffman (linux)。

* 建树模式下 (-m c)，输入字符权重文件，保存模型到文件。在屏幕中输出码表大小、WPL值。
* 编码模式下 (-m e)，输入文本文件（.txt格式），输出编码结果到文件
* 解码模式下 (-m d)，输入编码文件，输出解码结果到文件（.txt格式）

|  |
| --- |
| huffman.exe -m c -i input\_file -o model\_file # 建树模式  huffman.exe -m e -i input\_file -h model\_file -o output\_file # 编码模式  huffman.exe -m d -i input\_file -h code\_file -o output\_file # 解码模式 |

1. 主程序框架

|  |
| --- |
| int main(int argc, char \*argv[])  {  // Step 1. 解析参数，确定工作模式  …  // Step 2. 编/解码  if (mode ==‘c’) {  read\_weights(input\_file, weights);  build\_huffman(weights, model\_file);  save\_model(model\_file);  else {  read\_text(input\_file, data);  load\_model(model\_file, model);  if (mode ==‘e’) {  huffman\_encode(model, data, result);  }  else {  huffman\_decode (model, data, result);  }  write\_text(output\_file, result);  }  return 0;  } |

1